ECE 2574: Data Structures and Algorithms - Priority Queue

C. L. Wyatt

Today we will look a the priority queue ADT, its implementation in terms of SortedList, and its applications.

- Priority Queue ADT
- AbstractPriorityQueue
- Reusing SortedList
- Implementing using a linked list directly
- Applications of Priority Queues
- `std::priority_queue`

Recall the definition of a queue

A variation is the introduction of a priority: a priority queue or *heap*
Naive linked list or array implementations of a priority queue have O(n) performance for insert.
However a tree based implementation using arrays has O(log n) insertion (in a couple of weeks).

Priority Queue ADT

A priority queue is an object that contains items that have an associated priority.

- add (insert, push) adds an item to the heap
- remove (pop) removes the item with the largest priority
- peek (top) peek at the item that would be removed next
- isEmpty return true if the queue is empty

Warmup

Let a priority queue, q contain strings with associated priority values, k, with `q.push(k, string)` as the insert signature.
What are the contents of the queue after each operation? Assume the entry on the far left is the one to be removed next.

1. `Starting with an empty queue q`
2. `q.push(98,"A")`
3. `q.push(50,"B")`
4. `q.push(131,"C")`
5. `q.pop()`
6. `q.pop()`
7. `q.push(112,"D")`
8. `q.push(25,"E")`

An interface definition for priority queue

See `abstract_priority_queue.h`

Implementing AbstractPriorityQueue using SortedList

See `priority_queue.h`

Implementing AbstractPriorityQueue using a linked list directly

See `priority_linked_queue.h`

Example uses of priority queues

- Process Scheduling
- Network routing and switching (Quality of Service)
- Several algorithms in Artificial Intelligence (searching and planning)

The C++ standard library has a priority queue implementation.

```
#include <queue>
priority_queue<valuetype, container, compare> pq;
```

where:

- valuetype is the type of value being held
- container is the underlying container used (`std::vector<valuetype>` by default)
- compare is the functor used for comparison (`std::less<valuetype>` by default)

with methods:

```
void push( const T& value );
void pop();
const_reference top() const;
bool empty() const;
size_type size() const;
```

Example

Write a program using `std::priority_queue` to sort an array of random integers.

- create an array of 10 random integers
- sort the array using `std::priority_queue`

This is essentially the *heapsort* algorithm although it does not operate in place.

Next Actions and Reminders

- Read CH chapter 14
- Program 4 released. It is due 11/17.